

序列化调研报告

摘要：序列化指的是将任意的数据结构析构为一连串的字节。系统可以在另一个程序中通过这个字节流重组出一个等价的数据结构。

作者：王琦

日期：2015-01-02

1. Boost serialization

“

点击查看[官方文档](#)。

在boost.serialization中，我们使用术语“archive”来描述这种字节流的呈现方式("render")。这种呈现方式可以是二进制数据、文本数据、XML或其他用户选定的方式。

"output archive"和"output data stream"类似，下述代码通过<<或&操作符，实现了保存数据：

```
ar << data;  
ar & data;
```

"input archive"和"input data stream"类似，下述代码通过>>或&操作符，实现了装载数据：

```
ar >> data;  
ar & data;
```

当操作符作用于primitive data type时，数据即被简单的保存、装载；当其作用于class data type时，类的serialize函数被递归的调用，直至所有类中的数据都被保存、装载。

[官方文档](#)以bus_route, bus_stop, gps_position三个类为例，描述了：

- 需要修改类定义的序列化方法(intrusive)。
 - 实现方式：在私有方法中添加友元和实现serialize函数
 - friend class boost::serialization::access;
 - template void serialize(Archive & ar, const unsigned int version)
 - 示例：含有度、分、秒(degrees, minutes, seconds)的gps_position类
- 不需要修改类定义的序列化方法(intrusive)
 - 实现方式：
 - 要求类暴露出足够重构类状态的信息
 - 在类定义外重载serialize方法
- 类成员

- 当serialize作用于类对象时，该类的serialize函数将被递归调用
 - 示例：含有两个gps_position对象(latitude, longitude)的bus_stop类
- 派生类
 - 派生类的serialize函数发起基类的序列化，但**不要**直接调用基类的序列化方法，而应该通过下述方式调用
 - ar & boost::serialization::base_object<base_cls>(*this);
- 指针
 - output archive
 - 指针指向的对象被序列化，而不是指针的值
 - 一个指针被多次序列化时，它指向的对象只会被序列化一次
 - input archive
 - 装载时，一个新的对象将被创建，指针将指向这个新的对象
 - 对于先前被多次序列化的指针，第二个及其之后的指针的值，会等于第一个指针的值
 - 示例：bus_route类可以拥有多个bus_stops指针，不同的bus_route对象，可以拥有指向相同bus_stop的指针——想象交错的公交线路
- 数组
 - 数组可以直接被序列化
- STL集合
 - serialization库中包含了所有STL类的序列化方法

2. Thrift

“

点击查阅[官方文档](#)。

thrift是一个软件框架，用来进行可扩展且跨语言的服务的开发。它结合了功能强大的软件堆栈和代码生成引擎，以构建在 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml 这些编程语言间无缝结合的、高效的服务。

它支持多种语言之间的RPC方式的通信：php语言client可以构造一个对象，调用相应的服务方法来调用java语言的服务，跨越语言的C/S RPC调用。

想要运行thrift, 需要：

1. 下载Apache Thrift
2. 构建并安装Apache Thrift编译器
3. 编写.thrift文件，thrift编译器将根据你的.thrift文件生成源代码
4. 执行thrift --gen <language> <Thrift filename>

未完待续...

3. Protocol buffer

Protocol buffer是一种用于序列化数据结构的语言中立、平台中立的可拓展机制。

以C++为例.

未完待续...